

Kuwaiba Open Network Inventory

Version 0.6 [02.24.2014]
User Manual

Please visit <http://www.kuwaiba.org> for documentation, latest updates and upcoming events

Change History

Responsible	Description	Date	Reviewed by
Charles Bedón charles.bedon@kuwaiba.org	First issue shipped with version 0.1.1.	September 28 th 2010	
Charles Bedón charles.bedon@kuwaiba.org	Update to cover the new features in version 0.2	November 26 th 2010	
Charles Bedón charles.bedon@kuwaiba.org	Update to cover the bugfixes and changes in version 0.2.1	December 26 th 2010	
Charles Bedón charles.bedon@kuwaiba.org	Changes in version 0.3 alpha	January 18 th 2011	
Charles Bedón charles.bedon@kuwaiba.org	Changes in version 0.3 beta	February 3 rd 2011	
Charles Bedón charles.bedon@kuwaiba.org	Changes in version 0.3 beta 2	March 13 th 2011	
Charles Bedón charles.bedon@kuwaiba.org	Changes in version 0.3 stable (the clear button in the graphical query editor)	May 16 th 2011	
Charles Bedón charles.bedon@kuwaiba.org	Adapted to version 0.4	May 23 rd 2012	Adrián Martínez adrian.martinez@kuwaiba.org
Adrián Martínez adrian.martinez@kuwaiba.org	Adapted to version 0.5	Oct 23 rd 2012	Charles Bedón charles.bedon@kuwaiba.org
Charles Bedón charles.bedon@kuwaiba.org	Added documentation for Pools module	Jun 4 th 2013	Adrián Martínez adrian.martinez@kuwaiba.org
Charles Bedón charles.bedon@kuwaiba.org	Added documentation for Data model Manager module and some other minor changes	Jun 12 th 2013	Adrián Martínez adrian.martinez@kuwaiba.org
Charles Bedón charles.bedon@kuwaiba.org	Updated to version 0.6	Feb 24 th 2014	Adrián Martínez adrian.martinez@kuwaiba.org

License



This document is published under the terms of a license Creative Commons by-nc-sa. You can find details about it at

<http://creativecommons.org/licenses/by-nc-sa/2.0/>



Kuwaiba Server and Client are licensed under EPL v1. You can find the whole text of this license at

<http://www.eclipse.org/legal/epl-v10.html>

Disclaimer

- Netbeans and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners. The Kuwaiba project is not endorsed in any way to any of them.
- This document is provided “as is”, with no warranty at all. Install the software and follow the instructions included at your own risk.
- Kuwaiba uses third-party components licensed under compatible licenses (LGPL, BSD-like). You can find a complete list at the project's web page.

Chapter 1. Network Inventory Management in a nutshell

When you have a lot of things is easy to lose some of them and it's even easier to end up with a monumental mess if you're not organized enough. As a result, you will find yourself buying something you already have but you haven't managed to find, or wasting precious resources (money or space) storing stuff you barely know what is for.

The same happens to the IT and telecommunications infrastructure (actually in nearly every business handling assets [1]). It tends to be complex and its components lose visibility as the base of users/subscribers/customers increases. So it's common to see servers, workstations, antennas and equipment of all species piled in some dark room. In the mean time, sysadmin or technical guy's mind apart, it's not possible to find the updated information about the operative elements configuration or even location, the rack space is wasted and the patch panels are a land of no one, so how will anyone know which cable to pull?.

They say you can't manage what you can't control, and you can't control what you don't measure, and that's more or less the point of all this inventory thing. If you have under control what you have and the way everything is related will be more likely to achieve the goals of the Operations Department or the IT Staff, if they use this information to attend the incidents effectively and plan the capacity the more accurately way possible, among other things.

Having said this, three possible types of resources may be identified:

- **Physical:** These are the equipment, pipes, cables, fiber optics, facilities, parts and in general every physical asset from a port to a building.
- **Logical:** These are all the resources related to non-tangible technology assets. In this group fits timeslots, virtual circuits, VLANs, disk space, available bandwidths, etc.
- **Administrative:** These are all those related to administrative tasks, human resources or services. Here you can find customers, services associated to those customers, SLAs (and related parameters like availability or throughput), account managers, technical staff assigned to those services, vendors, brands. In short, all not-operational resources related to a given service management workflow.

As you may guess (or have experienced), to have all this information updated and complete at a single point is a hard task. Mostly because IT and telecommunications are ever changing environments and your network configuration layout can shift overnight. Updating all the data manually is almost impossible, but fortunately, you can grab a good part of it from the network elements if they have proper management interfaces.

From an ITIL perspective [2], within the Service Transition process there's a key concept called "CMDB". In short, is a place where you have Configuration Items (CIs, any IT asset that can be configure or have configuration information) **and their relationships**. Yes, not only the objects. Indeed, multiplexers, radios, routers are not islands. They're connected

physically or logically, and people may be related to them as well. A network inventory system is a CMDB but may extend its functionality in order to provide workflow management, network designing features, etc. But the core of it is a Configuration Management Database.

But one thing is different from other scenarios is that technology assets may not be physical. If you have a core using SDH you need to know what VCs are free, which ones are protected and what kind of protections they have, but if you manage switches you may want to know the bit rate configured for a given port or how the VLANs are set up. At application level, you'd like to know what DNS are configured for a given Web Servers or where are the Domain Servers...

Kuwaiba Open Network Inventory aims to be a solution for this scenario. It is completely Object Oriented, so you can model your network according to your needs, no matter if you're an ISP, a carrier or just a guy with a large IT infrastructure. Remember, Kuwaiba is under active development and new models are added every release. Please request the ones you consider as top priority in [this thread](#) [3].

Chapter 2. The login window

The default user/password is **admin/kuwaiba**. When you execute the binary at \$CLIENTPATH/bin/kuwaibainventory (Unix-like systems) or \$CLIENTPATH/bin/kuwaibainventory.exe (Microsoft Windows® systems) you'll get a window like this (remember, you need JRE1.6 to run the client. JRE 1.7 is not currently supported. Check [here](#) [4] for details):

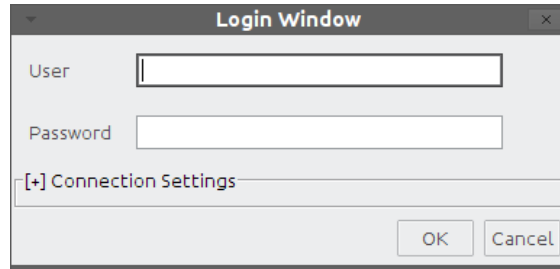


Figure 1: Login window with the Connection Settings panel collapsed

Pressing “Cancel” will close the application, and pressing “OK” will attempt to log into the system using the default connection values. This is, *localhost* as server, *8080* as server port and “/kuwaiba/KuwaibaService” as WSDL location. If you deployed the server with the default settings in the same box as the client, this should be enough.

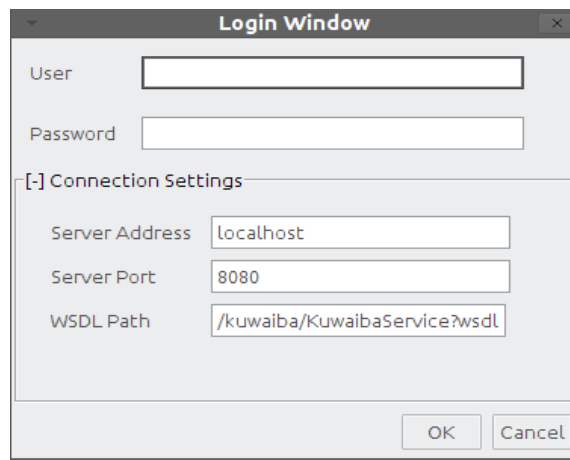


Figure 2: Login window with the Connection Settings panel expanded

You can expand the Connection Settings panel in case you want to change those values.

- **Server address:** Name or IP address where the server is running
- **Server port:** Port where the applications server is listening at.
- **WSDL Path:** Path to be appended to the server address and port to form the valid

URL to access the WSDL expose the methods to be consumed. It's important to point out that the Kuwaiba desktop client has a webservice client within, and that's the mechanism used to interact with the server. If you're not sure about this value, try typing in web browser an URL like this:

`http://<server_address>:<server_port><wsdl_location>`

An XML document must be displayed

If you were able to log in successfully, the first thing you will see is an empty window with only a toolbar at the top.

The last successful login settings (except the password) will be saved and used the next time you see the login window.

Troubleshooting: You can get a “Not enough privileges” message in case the user/password provided are invalid. If you get an “EJBException” that means an internal server error occurred, so you probably need to take a look at the server log for details, but most of the problems in this regard has to do with the database connection (the database server is not running, can't be reached or the credentials used to access it are invalid). Contact the application administrator if you get this error. A “HTTPServletException: Connection refused” indicates that the server is not running at all (or couldn't be found), check if the WSDL location is correct.



Figure 3: Main toolbar layout



Query Manager (Chapter 8)



Refreshes the current **focused** component



Refreshes the local cache. Use it whenever you make a change to the class metadata (i.e, when you change the icon or display name for a given class or when you add a new list type element). Also hit this button if you add a new list type item.



Basic view (Chapter 5)



Topology Manager (Chapter 6)



Audit Trail (Chapter 13)



User and Group Management (Chapter 12)



Data Model Manager (Chapter 2)



Containment Hierarchy Manager (Chapter 3)



List Manager (Chapter 7)



Navigation Tree (Chapter 4)



Pools Manager (Chapter 9)



Rack View (Chapter 10)



Service Manager (Chapter 11)

Chapter 2. Data Model Manager

One of the key features of Kuwaiba is that it is completely object oriented. It means that every business (Router, City, Port) and application (users, types) element is represented by an object in the application. Likewise, every attribute is a Field in a Class [5]. The set of classes, attributes and relationships between them is called data model. There's a default data model, but you can customize it depending on your needs by adding, removing and modifying classes. To do this, use the Data Model Manager module.

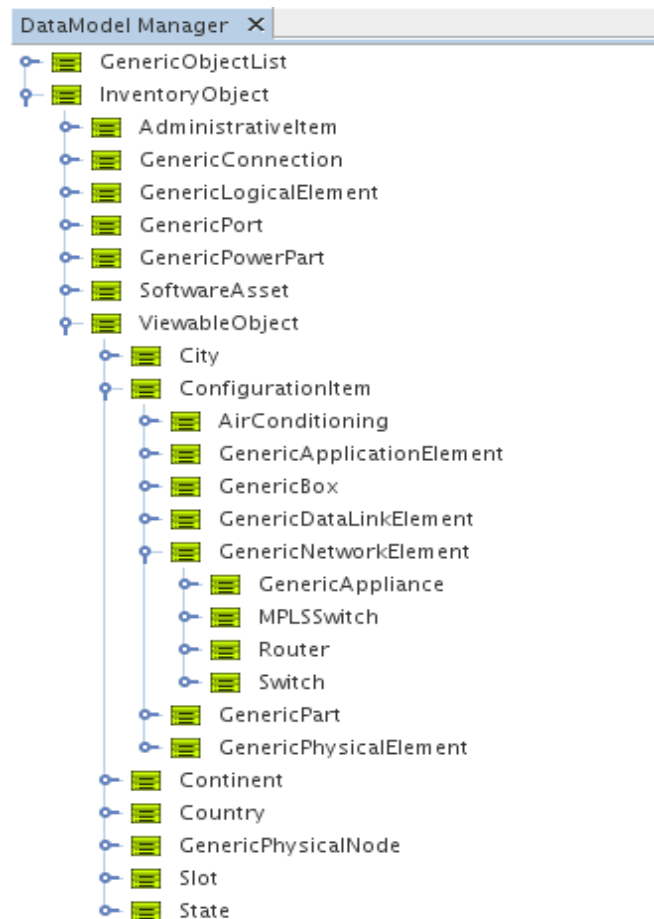


Figure 4: Data Model Manager main window

The main window shows the class hierarchy, at least from `InventoryObject` (its subclasses represent all the possible elements that are treated as inventory assets) and `GenericObjectList` (each subclass is a list type. See chapter 7 for details). To edit a class, open the Properties window (Window → Properties), and after selecting a node in the tree, you will see something like this:

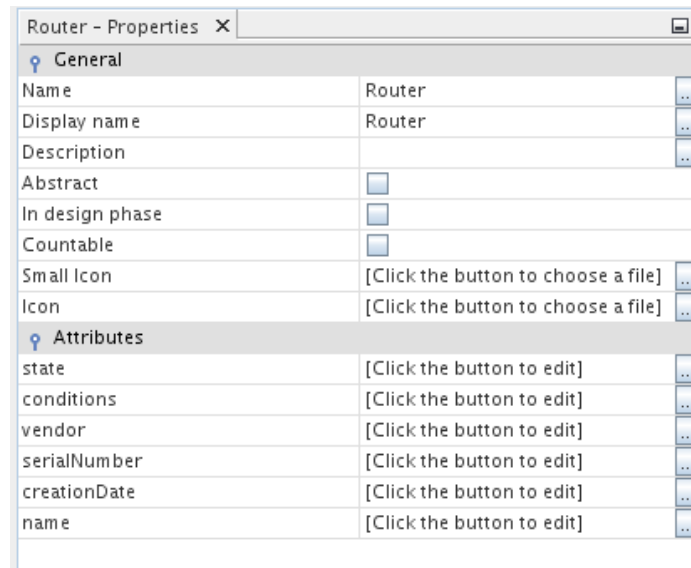


Figure 5: Class properties

The property sheet is divided in two sections: **General**, which contains the properties of the class itself, such as its name (can contain only letters and numbers with no special characters or blank spaces), the way the class name will be displayed everywhere else (useful for internationalization purposes), a description (useful to document the data model as it grows), if the class is abstract (abstract classes can not be instantiated, they're used to give consistency to the data model). Sometimes, the construction of an extension of the default data model takes time, use "In design phase" to mark a class as part of an ongoing work to extend the data model. Classes with this attribute set to true can't be instantiated. All new classes have it set to true by default. Countable is not being used currently, but it should be used to mark classes whose instances can have graphical views, but they're not really part of the inventory, such as Slots. The Small Icon is the icon displayed in trees (16x16 pixels), while the Icon is the image displayed in graphical views (32x32 pixels).

The second section is related to the class fields (attributes). In the figure above, class Router has six attributes: *name*, *state*, *conditions*, *vendor*, *serialNumber* and *creationDate*. Click the button next to the attribute name to customize it.

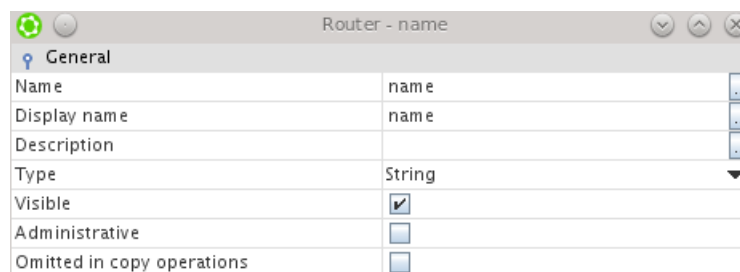


Figure 6: Editing an attribute

In this window, you can modify the attribute's name, display name, description, type (the drop down list will show you primitive types -String, Integer, Float, Long, etc- and available list types). When you change an attribute's type, all existing instances will be modified to reflect the change, which means that the values of the modified attribute will be converted to the new type if possible (say, from Integers to Strings). If the conversion is not possible, the new value will be set to null (**Warning:** You may lose data in this operation!). You can also manage the attribute visibility. Attributes marked as “Administrative” will be shown in a separate tab in the object's property sheet. Sometimes, there are attributes that are used only for administrative purposes and might confuse the end user if mixed with the regular attributes. Finally, you can choose what attributes shouldn't be transferred from one object to another in a copy operation.

Important: Although there's a cancel button on the bottom of the window, it does not really work (sorry, we're trying to find a workaround). When you perform a change, it's saved immediately.

You can create and delete classes and attribute by using the right click menu of every class node as seen in the picture below:

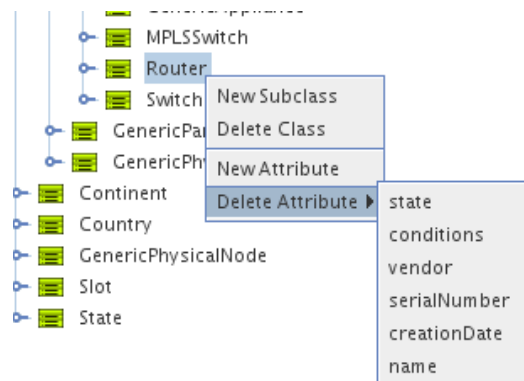


Figure 7: Right-click menu in class nodes

New subclasses inherit the parent class attributes. Classes with instances or subclasses can not be deleted (this is a design decision to avoid unintended loss of data). Attribute “name” can not be deleted.

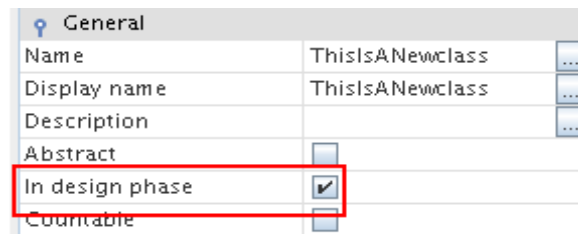
Note 1: This module replaces the Class Manager and Attribute Manager modules present until version 0.4.

Note 2: Classes provided in the default data model are called **core classes** and can not be deleted (though, they can be customized as any other class).

Important: It's highly recommended NOT to rename abstract core classes, since they're used internally in some cases and renaming them may affect some features..

Important: Before renaming classes, close the Navigation Tree window, if open.

Important: Modified classes can be used immediately. New classes will be marked as “in design” right after their creation. This is done to keep other users from using the classes until they're totally finished. Once you are satisfied with the new classes, uncheck the “In design phase” attribute as shown below.



The image shows a 'General' tab in a software interface. It contains a table with the following rows:

General	
Name	ThisIsANewClass
Display name	ThisIsANewClass
Description	
Abstract	☐
In design phase	☒
Countable	☐

The 'In design phase' row is highlighted with a red rectangle, and its checkbox is checked.

Figure 8: New classes are flagged as "In design phase" by default. While in that state, you won't be able to create objects of such class

Chapter 3. Containment Hierarchy Manager

Another key concept in Kuwaiba is containment. It consists of the ability to define what kind of objects can be created inside a given one. For example, a Country can be inside a Continent, but can't be inside a Rack. A port is usually within a board, and not inside a City. These business rules can be defined using the Containment Hierarchy Manager.

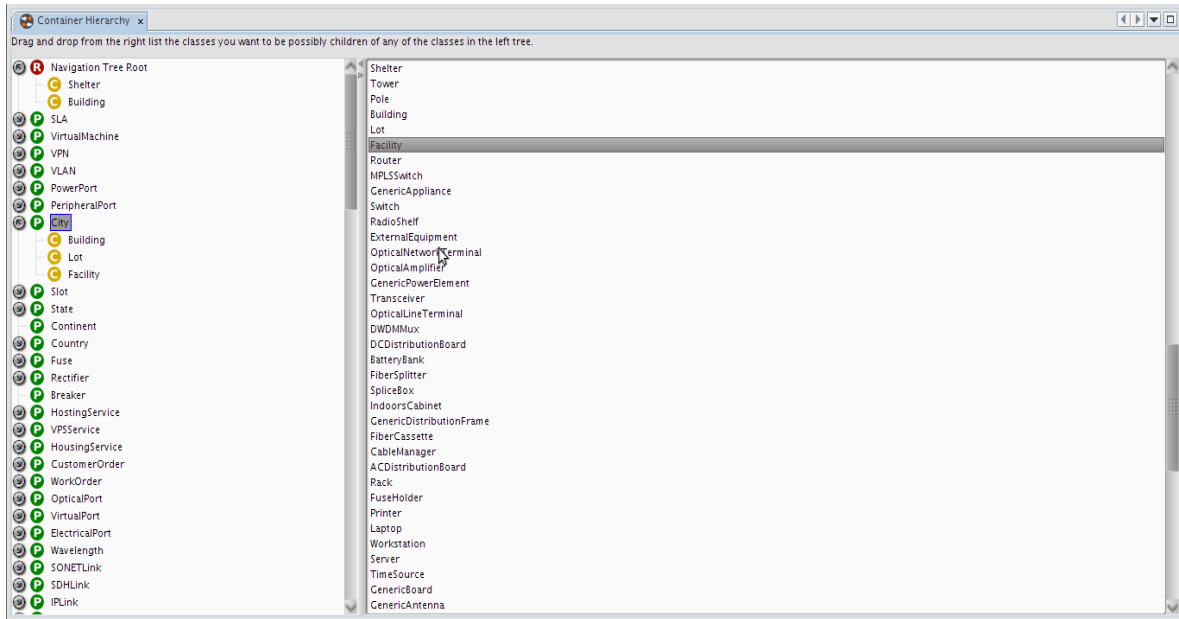


Figure 9: Containment Hierarchy Manager main window

The main window is divided in two panels. The one at the left is a tree that holds all the classes plus the Navigation Tree Root. The children of every node are the possible classes that can be contained. In the figure above we can see two nodes expanded: City, which has three nodes inside: Building, Lot and Facility. That means that below a given city you will only be able to add buildings, lots or facilities as direct children. If you don't care about buildings, just drag Room or whatever fits your needs from the list at the right side and drop it under the City node.

Within “Navigation Tree Root” we can find two children: Shelter and Building, which means that you can only add instances of these classes to the navigation tree root (the first level, marked with a star icon, as shown in the next chapter).

If you want to remove a children just right-click on it and select “Remove”, and instances from that class will no longer be available to be added under the parent class.

Shortcut: CTRL+H

Hint: Selecting any node in the tree and writing a text string will attempt to search for a

class or attribute starting with such string. If there are several matches, F3 will jump to the next one. The same applies to the list at the right panel.

Important: Containment hierarchy **is not** the same as Class hierarchy. It's a common mistake to confuse these terms. Containment hierarchy, as stated before is the way an object can contain in the real world another one (as in a box that holds small objects within). Class hierarchy is an object oriented programming concept (see *inheritance* [5][8]) that refers how code can be reused passing some code and attributes from a class to another (as the class Employee can inherit the attributes “name” and “gender” and the methods -logic- “walk” and “breath” from a class Person).

Chapter 4. The Navigation Tree

There's not too much to comment here, since it's a common feature in all systems.

This module is composed by two windows: The tree itself, which shows the business objects organized according to the hierarchy defined with the tool described in the previous chapter.

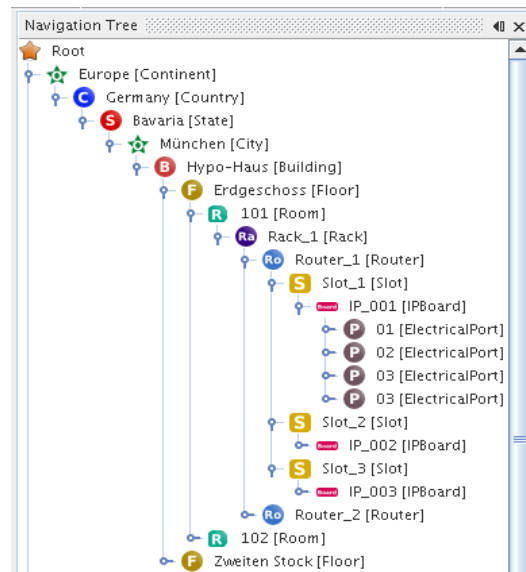


Figure 10: Navigation tree

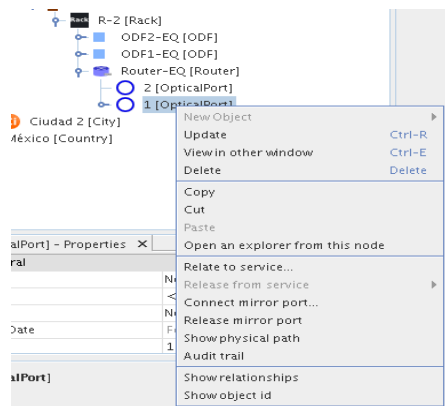


Figure 11: Navigation tree. Object options

Each node represents an object, and it has actions associated:

- **New:** Creates a new object of a given class, depending on what was configured at the Containment Hierarchy Manager.
- **Update:** Refreshes the node's information

- **View in other window:** The default Properties window updates itself depending on what node is selected. If you want a property window showing the object's information permanently, choose this option.
- **Delete:** Deletes the current object and any object below it (if you delete a Country, it will delete states, cities, buildings and any other contained element as well).
- **Copy/Cut/Paste:** Usual actions, only applicable if the target object allows it (you can't cut a Rack and paste it into a Continent if the containment was not configured to support such operation). **Important:** The CTRL+C and CTRL+V shortcuts was disabled in version 0.2.1 due to technical issues, but you can use the context menu to perform these actions. It's also possible to **move** an object by dragging and dropping the selected element or making a **copy** of an object by holding pressed the CTRL key while you drag and drop it. Multiple selection is supported.
- **Explorer from here:** Spawns another window showing a tree whose root is the selected node. Useful to explore places located deeply into the hierarchy, like ports.
- **Relate to Service:** See Service Manager (Chapter 11).
- **Release from Service:** See Service Manager (Chapter 11).
- **Connect Mirror Port:** Only available on ports. See Basic View (Chapter 5).
- **Release Mirror Port:** Only available on ports. See Basic View (Chapter 5).
- **Show Physical Path:** Only available on ports. See Basic View (Chapter 5).
- **Audit trail:** This option will show you the list of updates performed on that particular element. See Audit Trail (Chapter 13) for details.
- **Show relationships:** This will show you the special relationships to (and from) this object. This is useful to see how elements in a model are related. For example, if you select this option on a port that is connected to a cable/fiber, you will be able to see it (endpointA and endpointB relationships) or if that port is related to a service, you will see it under the label "uses".
- **Show id:** This option will show you the object id (useful to make queries) and the position of the object in the containment hierarchy. You can copy all that text by selecting it and pressing CTRL+C.

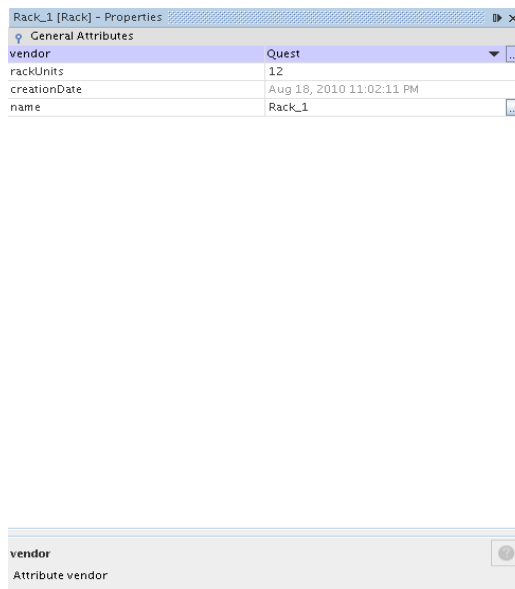


Figure 12: Properties Window

The other window is the Properties Window as shown in the figure above. This one displays the visible attributes and lets you edit them. Every change is automatically committed to the database once you hit “Enter”.

Important: Properties window is not opened automatically with the navigation tree. You have to open it by selecting the menu Windows → Properties.

Shortcut: CTRL+N

Hint: The navigation tree supports in-line editing. If you hit F2 when a single node is selected, you will be able to edit the object's name

Chapter 5. Basic View

This module is used to get presents a graphical representation of how elements are connected physically. The most basic view shows by default all direct children and the connections between them. Note: Only instances of subclasses of ViewableObject can use this module (i.e. ports don't have views). Please refer to the data model for details [9].

Before to show the general features in this module, it's important to make some concepts clear. This is a physical-only topology. It's only about cables, ports, etc. in order to support this scenario, the data model provides four entities (note: Although it's similar approach, the following concepts are not related to power connections, only communication ones):

Connections: These are all point-to-point links. In the current data model, there are three types of connections: ElectricalLink (for electrical connections like coaxial, twisted pairs and the like), OpticalLink (for fiber optics) and RadioLink (for radio links -WiFi, Microvave, WiMax, etc-).

Containers: All objects that can be used to contain, “conduct” and protect connections (understanding a “connection” as we defined above). There are two types of containers: WireContainer (used to contain all kind of cables -wires and fibers-) and WirelessContainer (used to contain radio channels or any other non-tangible medium).

Nodes: These are all those objects were a container can end. For this release there are seven possible classes: Tower, Warehouse, Facility, Shelter, Building, Floor and Room (in the data model context, all these are somehow subclasses of GenericPhysicalNode)

Endpoints: These are all those objects where a connection can end. In practice, they're all kind of ports (in the data model context, all these are somehow subclasses of GenericPort).

Important: You can only connect nodes using containers and endpoints using connections. In order to use this module is necessary to have the Navigation Tree already opened, and every time a node is selected, the view for such element will be shown in the canvas. If you haven't saved previously a view for an object, the default place all children in a single line with a padding of 100 pixels between each child.

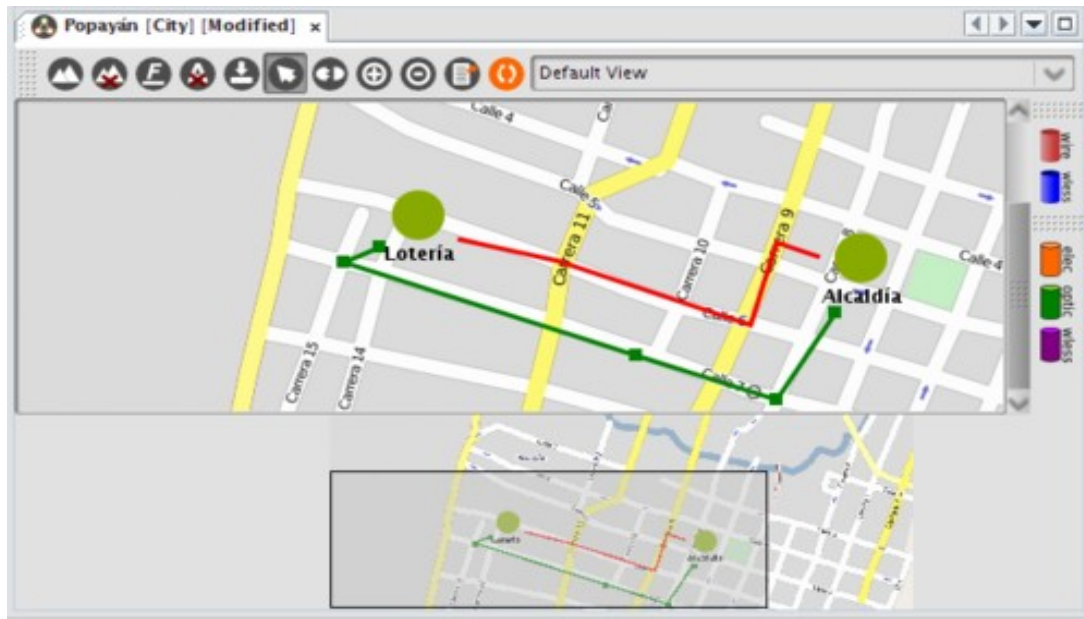


Figure 13: Custom view for a demo city (the background is an image from GoogleMaps). It has two wired containers (red) and a wireless one (blue)

The figure above shows a custom view for a demo city. As this is an example, the container hierarchy defined has been simplified as well (Root → City → Building → OpticalPort)

	Add background
	Remove background
	Format text (font type, size and color for labels)
	Show/hide the node labels
	Save view
	Select tool (used to move and select objects, it's selected by default)
	Connect tool. Used to connect elements. Whenever selected, the “WireContainer” button is pushed automatically by default.
	Zoom in
	Zoom Out
	Refresh view
	Explore Connections
	Available views for the current object. By now, only the default view is listed. Future releases will include extra views.
	Export view to JPG/PNG formats



Select it to create a WireContainer between two Nodes (A node is, technically speaking, any instance of a subclass of GenericPhysicalNode. See [9] for details. In practical terms, a Node is a Building, a Room, a External Cabinet, a House, a Manhole, etc. Anything that can be the endpoint to a pipe containing cables). This option is selected by default.

Select it to create a WirelessContainer between two Nodes

Select it to create an ElectricalLink between two endpoints (ports)

Select it to create an OpticalLink between two endpoints (ports)

Select it to create a WirelessLink between two endpoints (ports)

Example: If you want to connect two ports located in equipment inside different buildings using a cable (this is, an electrical transmission medium), press , then  and finally click on the source building and drag the cursor to the target building as shown in figure 12:

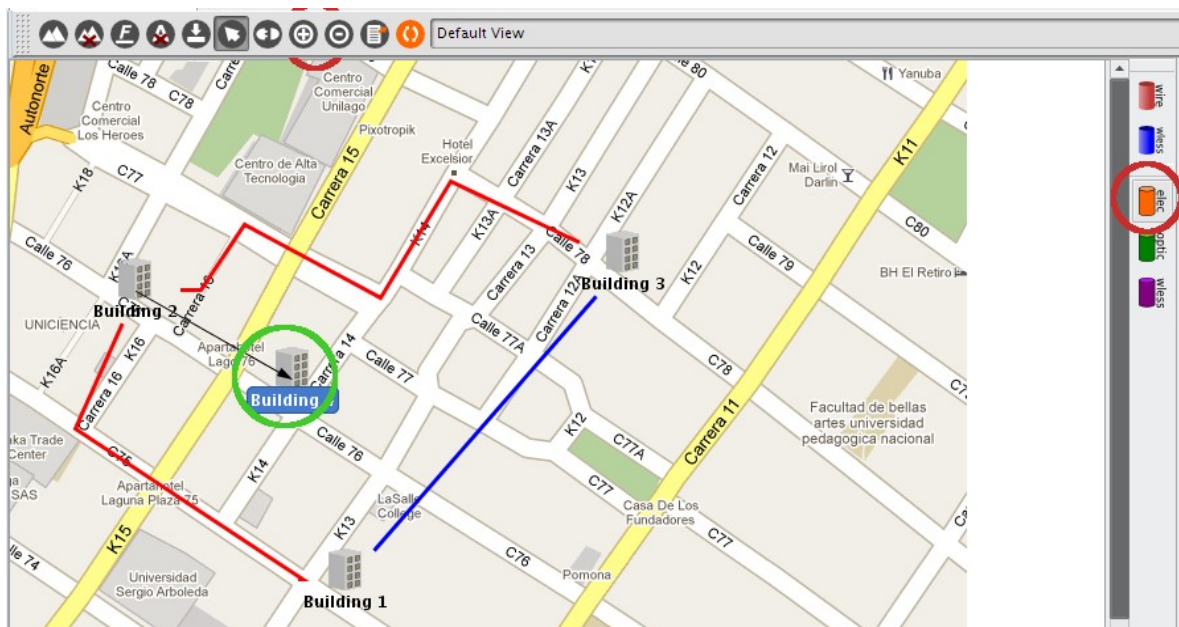


Figure 14: Creating an electrical link

This will launch a wizard where you can select the port you want to connect and set the properties of the new object.

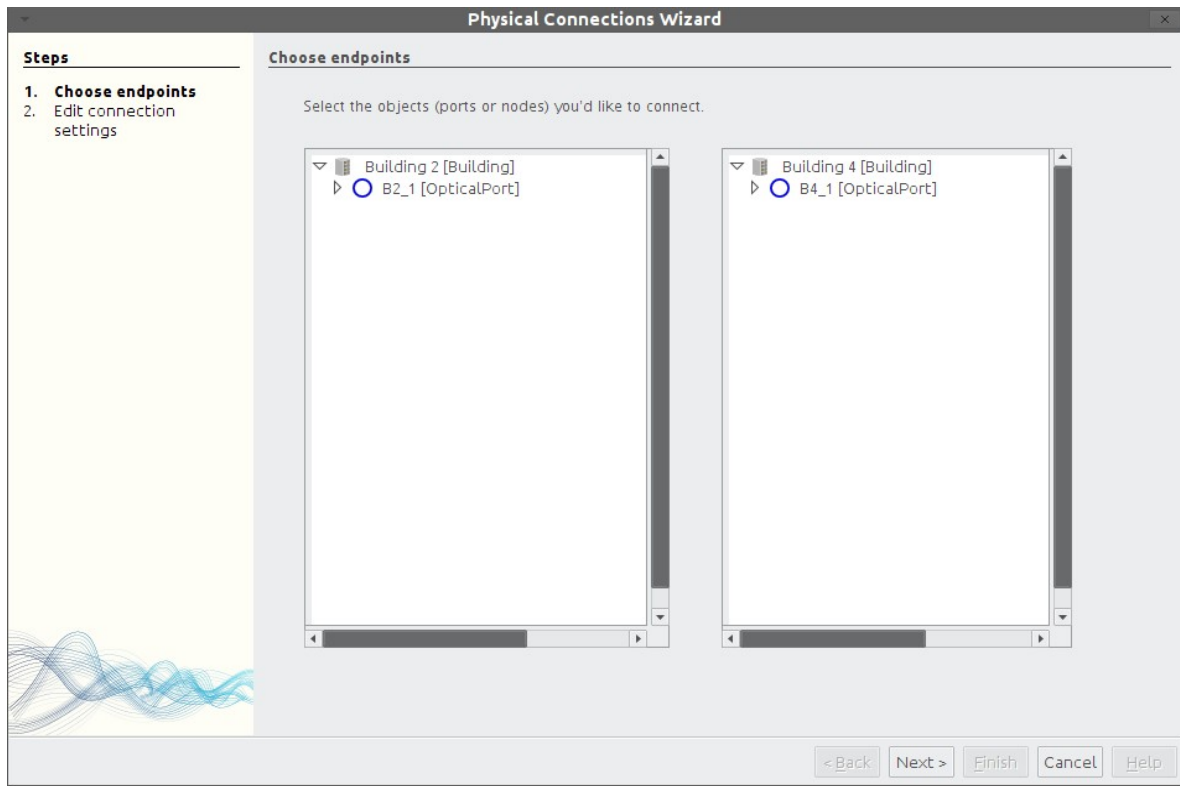


Figure 15: Wizard to create an electrical link

You will get warnings if you try to connect the wrong elements. So far port mappings are not supported, so you can, for example, connect an electrical port with an optical link (that is, a fiber). This kind of constraints will be added in future releases.

In the second step, you will be prompted for the basic information related to the new object to be created. These are the name and type. The type is a list that you customize in the List Type Manager (See Chapter 7). Just add whatever you need to the Electrical/Optical/WirelessLinkType and Wire/WirelessContainerType list (see figure below).

Note: If the item you add is not shown in the drop down list when creating the connection, you may need to refresh the local cache ( button).

The wizard will look like this:

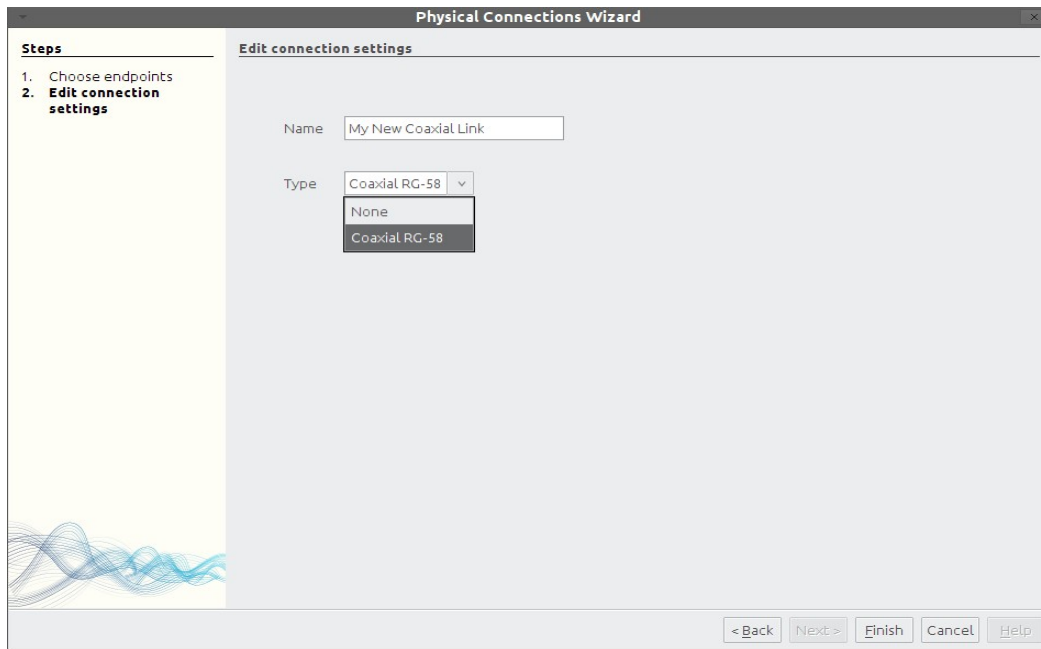


Figure 16: Step two: Basic information

Important: This kind of connection is typical in inside plant scenarios (you connect ports directly using cables or fibers), but in outside plant, you put a bunch of cables in a pipe and leave the cables endpoints open to be connected when needed. In these cases, you should create first the container (pipe), add the links (cables/fibers) and then connect them. To do that, you should choose the first icon in the toolbar on the right (the red cylinder) and connect two nodes (the endpoints to the pipe, Buildings, Houses, Outdoor cabinets), choose the type of container, what kind of links (fibers/cables) you want to put inside and how many of them will be created (you can add/remove more later). The step above would look like this instead:

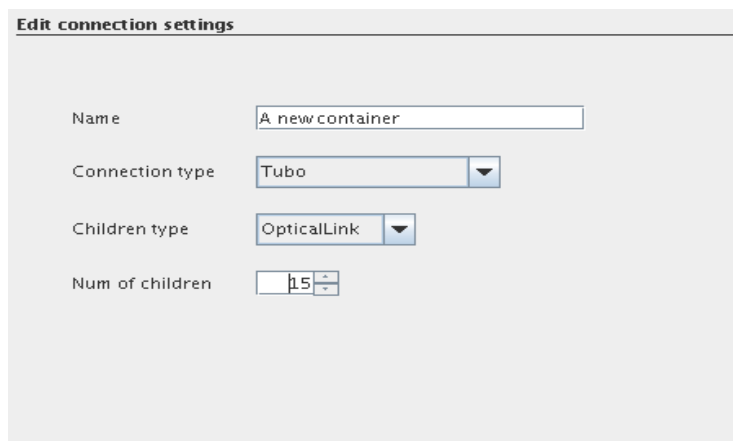


Figure 17: Step 2 of a WireContainer creation wizard. Its type is "Tubo" and it will contain 15 OpticalLinks (fibers)

Then the action is confirmed and the view is updated:

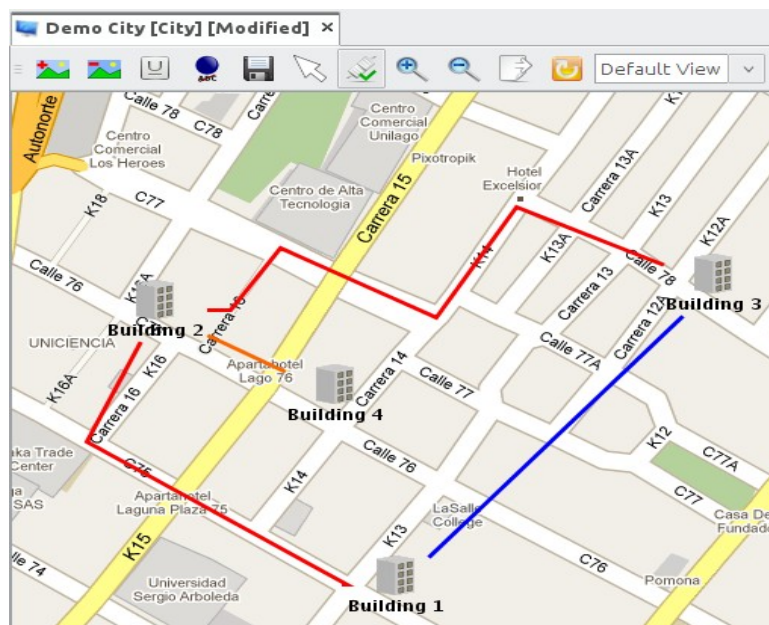


Figure 18: Connection successfully created (orange)

It's also possible to add and remove control points to the connections by double-clicking the line. You can format the text displayed using the format button . The changes will be persisted across sessions for all views.

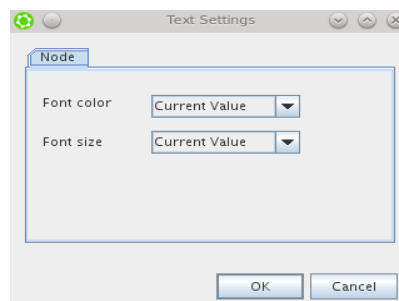


Figure 19: Text format options



Figure 20: Changing the text format

So, this is how a graphical view of an object (in this case a generic Facility) might look:

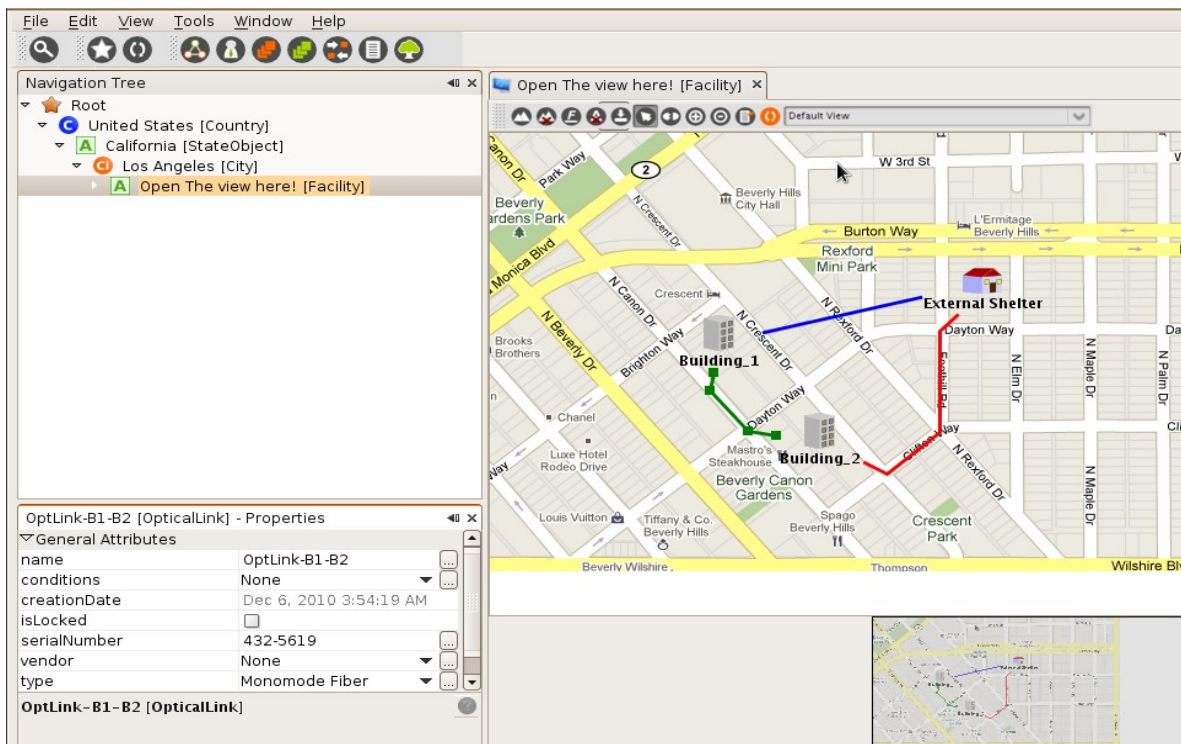


Figure 21: A simple view with paths and control points

To see the detailed information about an object within the view you just have to select it using a single click, and it will be shown in the Properties Window. Once selected, you'll see the current control points. To add a new one double click the line representing the connection and move it as you wish. To remove a control point double click on the desired

node as well.

Remember that every change in the connection path will mark the view as unsaved and you'll have to save it manually.

If you want to explore what's inside a container or a link (a container has links within and an OpticalLink may contain Wavelengths), use the tool  in the Basic View toolbar. Every time you select a connection, the tree below will be updated with the corresponding information:

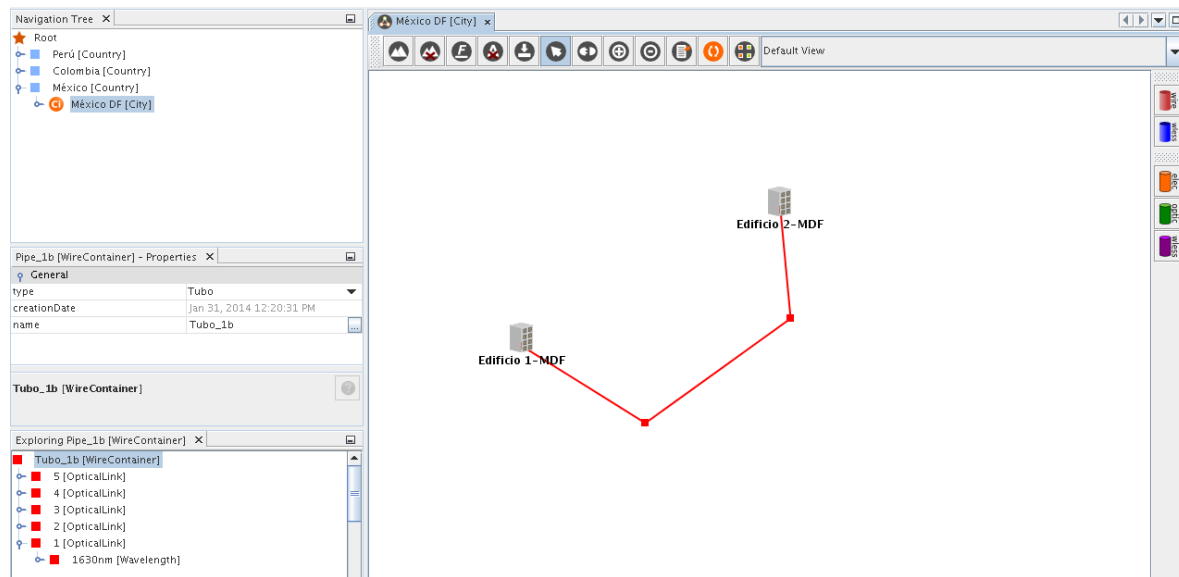


Figure 22: Connections Explorer (bottom left), Property Sheet (left), Navigation Tree (top left), Basic View (center and right)

The connection explorer will allow you to add and manipulate objects like you do in the Navigation Tree.

Additional to the type of connections described above, Kuwaiba also supports backplane/patch cable connections. This kind of connections are short distance connections (the type of connections you'd see between ports in the same ODF/DDF, i.e. 1-Front-1-Back). In Kuwaiba the port connected directly to another in the same level is called “Mirror Port” (what you get in one port is the same to what you see in the other). To create Mirror Port connections, right click on a port and select “Connect Mirror Port”. Then, a small window will pop up, asking you to choose the *sibling* port to create the mirror connection as shown below:

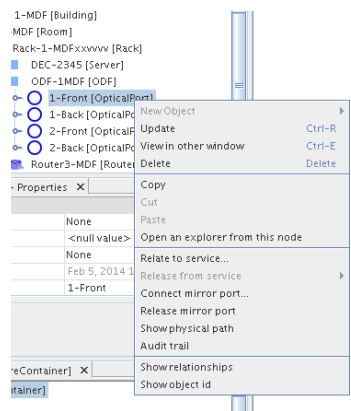


Figure 23: Port contextual menu on a port called 1-Front

If you choose (after creating the connection) the option **Show relationships** on the port, a relationship called “mirror” must be listed:

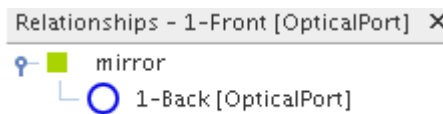


Figure 25: Special relationships on port 1-Front

You can chain many physical connections to build a path across passive equipment such as distribution frames composed by links and mirror relationships. To see the whole path, just right-click on any of the ports involved in the path and click on **Show Physical path**. A list like the one below will be shown. The element highlighted red is the one you are stepping on.

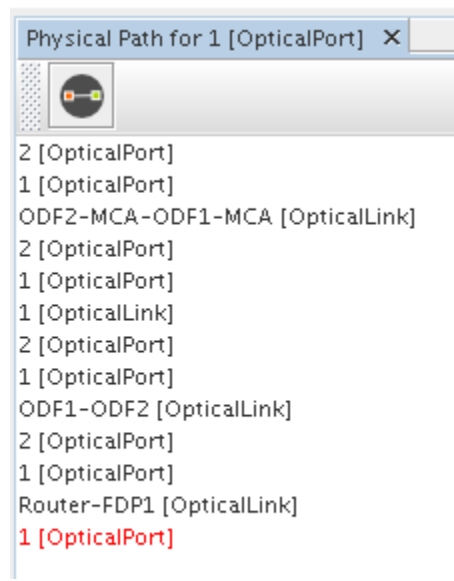


Figure 26: Sample physical path

If you prefer a graphical representation, just click on the button  and the path, along with the containment hierarchy of every component will be displayed.

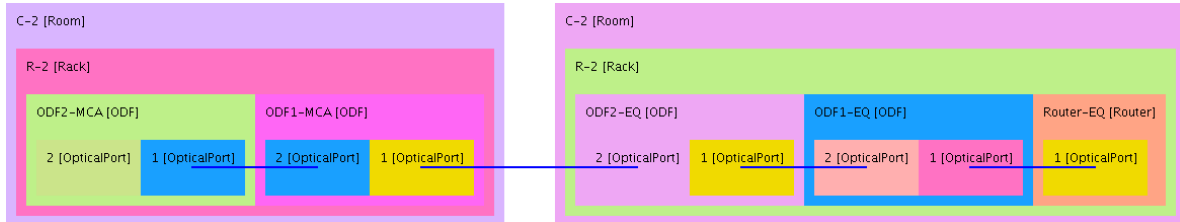


Figure 27: Graphical representation of a sample phsysica path

This representation can be exported to PNG/&JPG images.

Shortcut: CTRL+SHIFT+W

Hint: The object physical view supports in-line editing. If you double click a single node, you will be able to edit the object's name.

Hint 2: If you create/delete an object in the navigation tree, you must refresh the view

(button ) to see the new nodes.

Next versions: Future versions will include geolocation-aware views.

Chapter 6. Logical Topology Manager

This module is used to get a graphical representation of how elements are connected logically.

Note: Only instances of subclasses of ViewableObject can use this module (i.e. ports don't have views). Please refer to the data model for details [9].

	Creates a new topology view
	Opens a previously saved topology view
	Saves the current topology view. The topology view are stored in using an XML format [6]
	Deletes a saved topology view or clean the canvas is it's an unsaved one.
	Exports view to JPG/PNG formats
	Selects tool (used to move and select objects, it's selected by default)
	Shows/hides the node labels
	Connect tool. Used to connect elements. Whenever selected, the “WireContainer” button is pushed automatically by default.
	Inserts a cloud image
	Inserts a free frame with title to group elements
	Inserts a free editable label

Example: Design a topology view

You can drag and drop elements from the navigation tree into the topology view workspace, after this the connections can be done, because this a logical view there are no restrictions in which element could be connect with other. Until now there is no support for explicit port connections.

There are extra elements to improve the topology design, there are free editable labels, free frames with title to group elements, and a general cloud icon to represent any third-party provider.

Every element in the topology view design can be deleted making right click on it and selecting the delete option.

To see the detailed information about an object within the view you just have to select it using a single click, and it will be shown in the Properties Window.

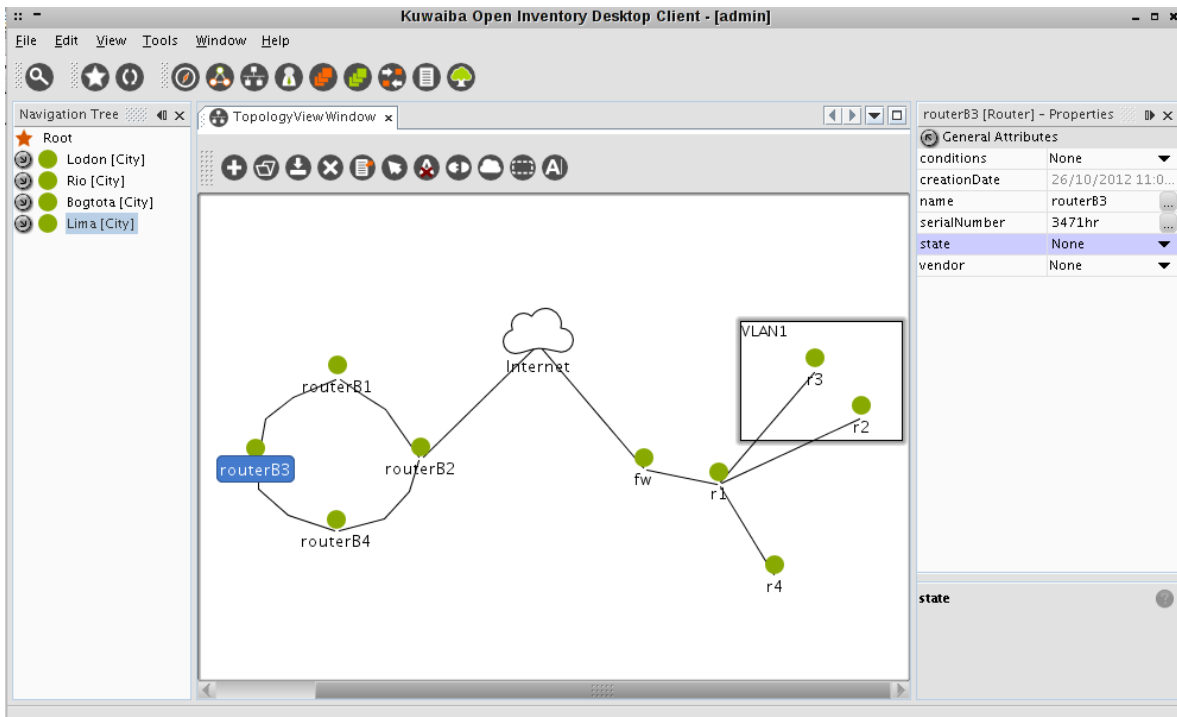


Figure 28: A sample topology

Chapter 7. Lists Manager

As commented in the chapter “Attribute Manager”, some attributes have a list type. The actual value is picked from a list of items. This is very usual when dealing with vendors or models.

You can create new list items by using the module called “List Manager”. This module consists of a simple navigation tree similar to the one seen in the last chapter.

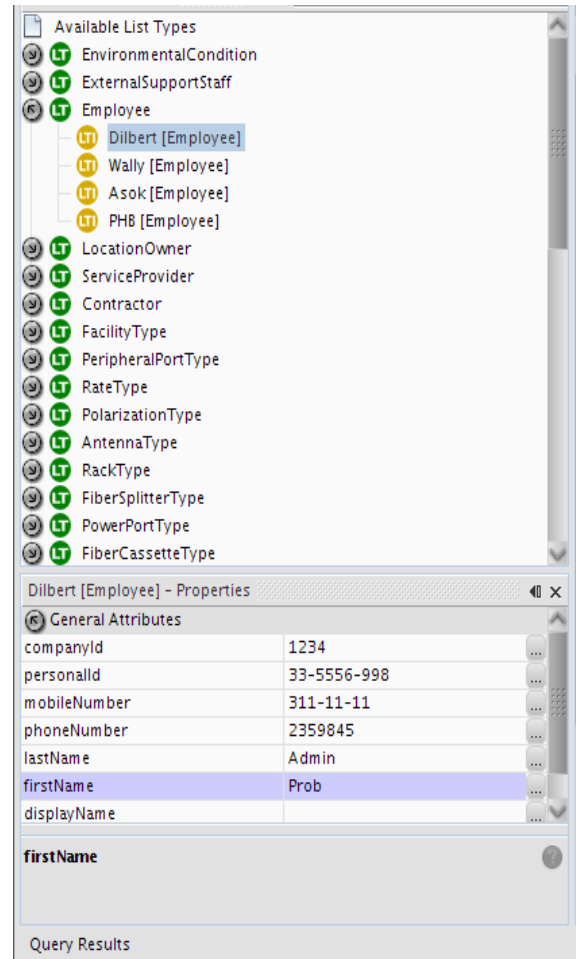


Figure 29: List Manager main window

By right-clicking and choosing “New” over a selected list type, you can create new items. The details for every item can be edited using the standard Properties Window as seen in the past chapter.

If the changes are not immediately reflected when editing an object, use the refresh cache button and update the object (Right click on the object node → Update).

Shortcut: CTRL+L

Hint: Selecting any node in the tree and writing a text string will attempt to search for a class or attribute starting with such string. If there are several matches, F3 will jump to the next one. The same applies to the list at the right panel.

Note: You may need to refresh the local cache ( button) if you change the name of any item so the change can take effect for already opened objects.

Chapter 8. Query Manager

Graphical Query Editor

This is the new supported method to design and perform queries. It has a graphical interface, based on nodes to represent the search criteria and connections to express the relationships. Once selected, you will get a blank canvas with a toolbar on top with the following elements:



Figure 30: Graphical query editor general toolbar

- | | |
|---|---|
| Search for | Select the element class you want to search for. In this list you're going to find only business-related classes (network elements, cables, etc), not system classes (users, sessions, etc) |
|  | Open a previously saved query |
|  | Save the current query. The queries are stored in the database using an XML format [6] |
|  | Deletes a saved query or clean the canvas is it's an unsaved one |
|  | Cleans the current query (clear completely the canvas for temporal queries or clear all nodes but the root for saved queries) |
|  | Edit details about the current query (name, description and owner) |
|  | Organize the node automatically |
|  | Executes the current query. |
| Logical Connector | The logical connector to be used to chain the query predicates. The default connector is AND. |
| Result limit | Max number of results per page. The default value is 10. Only integer values are allowed. |

When you select a class from the drop-down box, a graphical representation of it will be placed on the canvas

Search for Router

Router		
☐		name
☐		creationDate
☐		state
☐		serialNumber
☐		conditions
☐		vendor

Figure 31: Node added after selecting a class

In the example above the class Router was selected, and the corresponding node is created. You can search for a wider range of elements if you select an abstract class (often called GenericSomething). In the example above will search for all Routers in the database, but if you choose GenericPhysicalNode it will search for all physical nodes, this is, all objects that can be connected using a container (see Chapter 7 Object Physical View for more information about containers). That includes: buildings, rooms, floors, towers, shelters and facilities. Note that the root class node is colored green

Note: If you want to see all objects in the database (at least all relevant to the inventory) search for InventoryObject, which is the root of all classes related to the inventory. To see the complete hierarchy see related javadoc in the final chapter [7].

Back to the original example, if you just leave the router without modifications and execute the query , it will search **for all** routers available in the database and it will show only a column with the object's display name

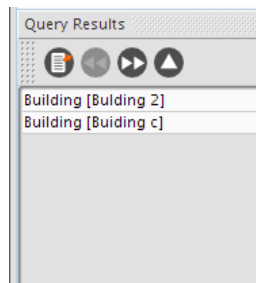


Figure 32: Query results for all routers

The window above is shown at the bottom of the main window and also has a tool bar with the following buttons:

-  Export the results to popular formats like CSV, XML, ODT and DOC (Only operational for XML and CSV so far)
-  Previous page
-  Next page
-  All results

Now, if we want to see more fields related to the found elements, we should use the “field visibility icons”. They’re those eyes you can see along with the attributes names (). By default none of them are selected, which means that no additional fields will be shown in the result list but the objects themselves. Think of them like the parts of a conventional SQL sentence that follows the SELECT clause:

SELECT creationDate, serialNumber FROM Router

You can obtain a query like this pushing the toggle buttons with the eyes on it beside “serialNumber” and “creationDate” just like the figure below

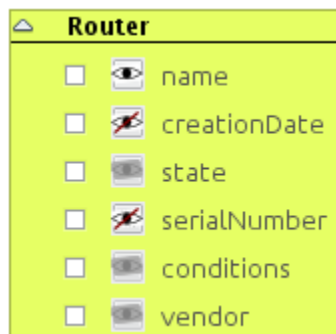


Figure 33: creationDate and serialNumber to be shown in the result list

When selected, the visibility icon turns into an eye crossed by a red line (🚫). If you don't want this attribute to be shown in the result list, press it again. This is what you will see when you execute the query:



	Router.creationDate	Router.serialNumber
BorderRouter_1 [Router]	Sun Mar 13 17:16:15 GMT-05:00 2011	BR987-654
BorderRouter_2 [Router]	Sun Mar 13 17:16:20 GMT-05:00 2011	BR123-456
SimpleRouter_1 [Router]	Sun Mar 13 17:16:45 GMT-05:00 2011	987-432
SimpleRouter_2 [Router]	Sun Mar 13 17:16:45 GMT-05:00 2011	123-678

Figure 34: Result set showing some fields

Now you may be wondering why the attributes “vendor”, “conditions” and “state” have their visibility icons disabled. That's because they're list type attributes (See Chapter 8. List Manager for details about list type attributes). Making again a comparison SQL, these attributes are related to other “tables” (other classes, actually). You have two ways to use these list type attributes as criteria: the first is just selecting the attribute's checkbox and choosing from a list the value that you'd like to use:

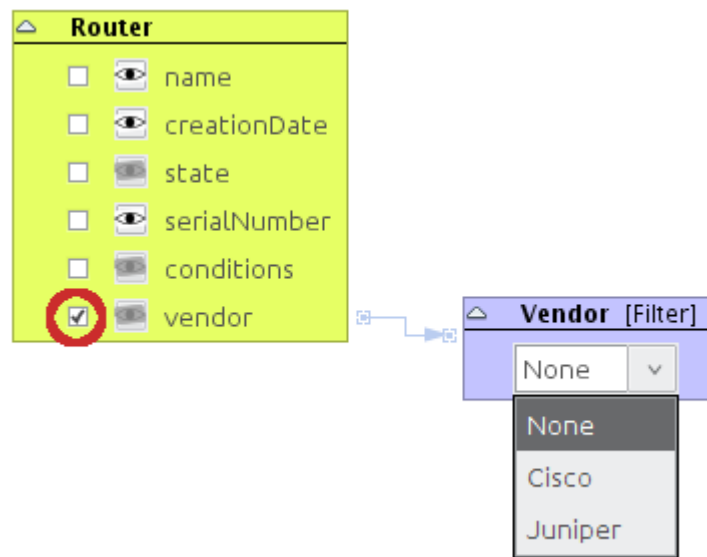


Figure 35: Using a list type attribute as filter

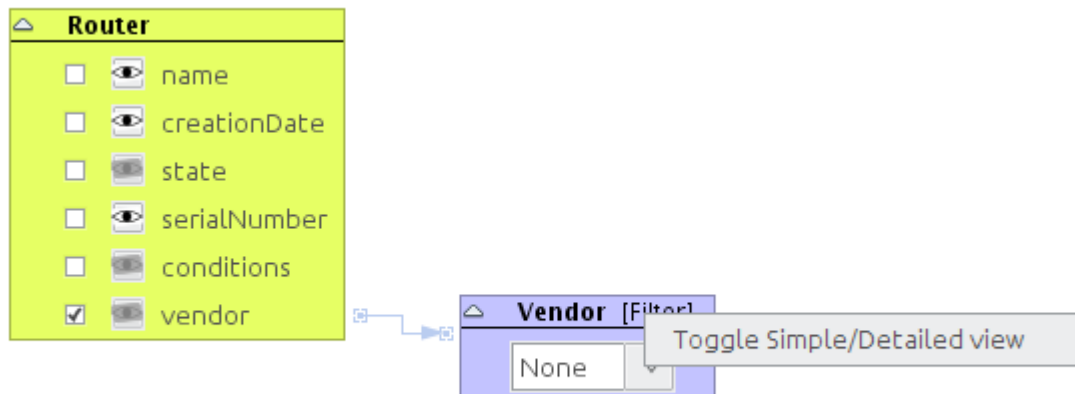


Figure 36: Changing the detail level on a class filter

Or you can pick manually what fields belonging to the relationship must be included in the result set. To do this, you first have to toggle the simple class node filter (in our example labeled as “Vendor [Filter]”) to a detailed view. You can do by right-clicking on the node's header and choosing the option “Toggle Simple/Detailed view”. The result is a node similar to the root one, but colored light orange (■):

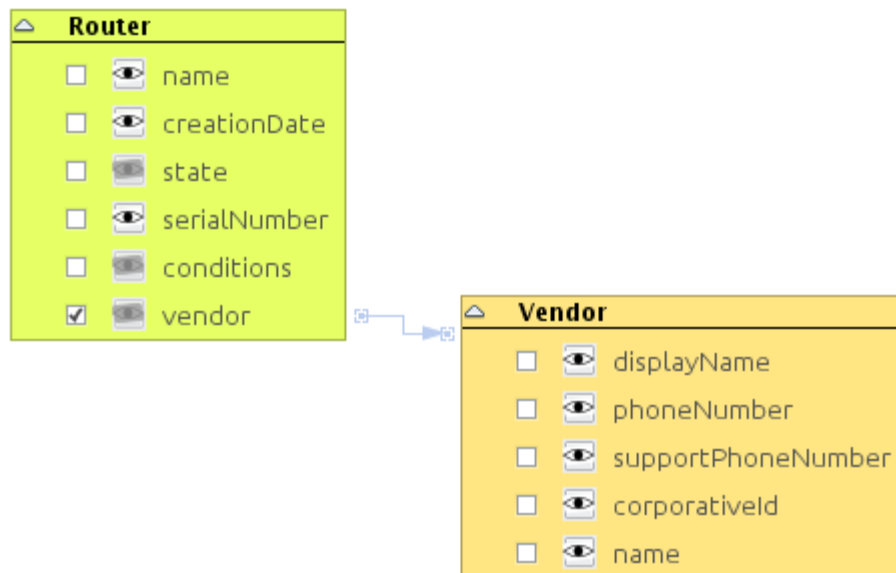


Figure 37: Expanded class node filter

You can toggle the node to the original fashion by repeating the procedure.

Example: We want to search for all routers and we want to see the fields **creation date**,

serial number, vendor and operational state. That would be:

1. Enabling the visibility icon for the attribute “creationDate”
2. Checking the checkbox in the attributes “vendor” and state and expanding the resulting nodes (we'll talk about the checkbox later in this chapter)
3. Enabling the visibility icon for the attribute “name” in both newly spawned nodes.

Let's see this in a picture:

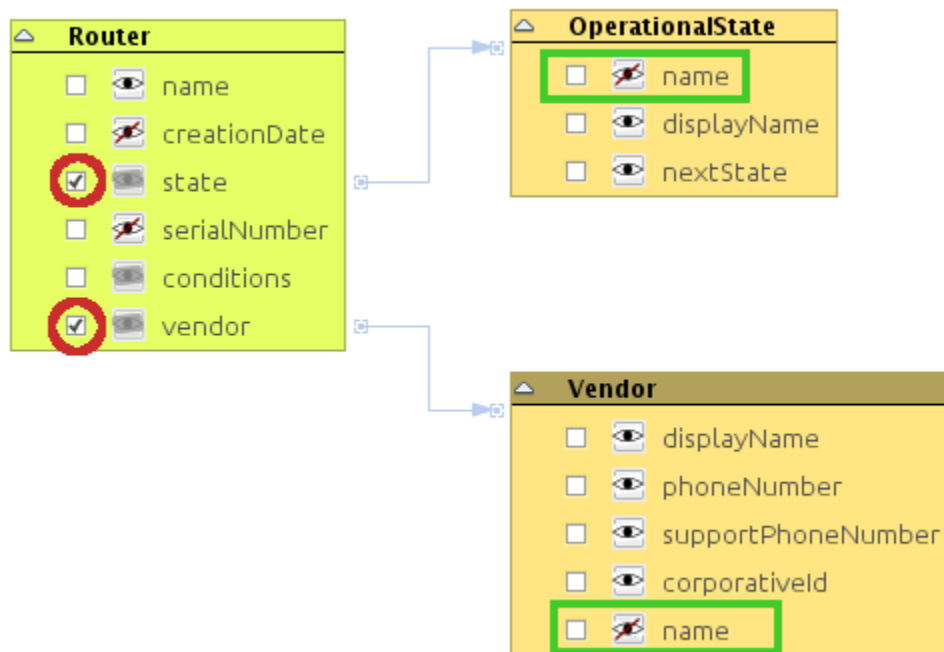


Figure 38: Simple query showing three fields from different classes

Resulting in a set like this:

	Router.creationDate	Router.serialNumber	OperationalState.name	Vendor.name
BorderRouter_1 [Router]	Sun Mar 13 17:16:15 GMT-05:00 2011	BR987-654	Working	Cisco
BorderRouter_2 [Router]	Sun Mar 13 17:16:20 GMT-05:00 2011	BR123-456	Working	Cisco
SimpleRouter_1 [Router]	Sun Mar 13 17:16:45 GMT-05:00 2011	987-432	Not Working	Juniper
SimpleRouter_2 [Router]	Sun Mar 13 17:16:45 GMT-05:00 2011	123-678	Being repaired	Juniper

Figure 39: Result set for the last query

Note: The operational states and vendors were created using the List Manager (under OperationalState and Vendor, respectively) and their properties set in the Properties window.

The question now is “what are those checkboxes for?” Well, they're used to construct the query predicates. Every time you select one of them, a new node is spawned. If the attribute is a basic one (String, Integer, Float, Boolean or Date), you will get a node like these

(otherwise you will get the class nodes seen above):

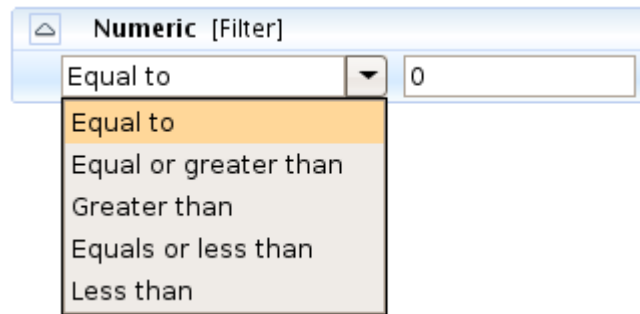


Figure 40: Filter for numeric values (integers and floats)

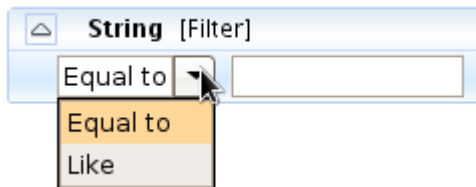


Figure 41: Filter for Strings

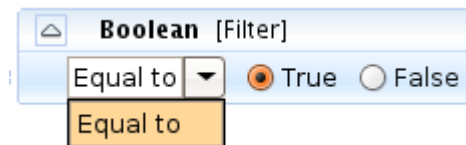


Figure 42: Filter for boolean values

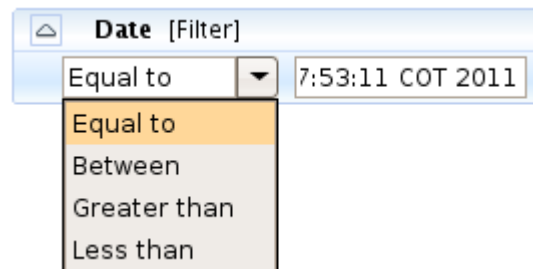


Figure 43: Filter for dates (not operational yet)

Example 2: We're going to search for Routers whose name contains the string “border” **and** the operational state **is equal to** “Working” **and** have Cisco as vendor. The fields to be shown must be “creationDate” and “operationalState”. Note that the result limit is set to **1** per page.

The query would look like this:

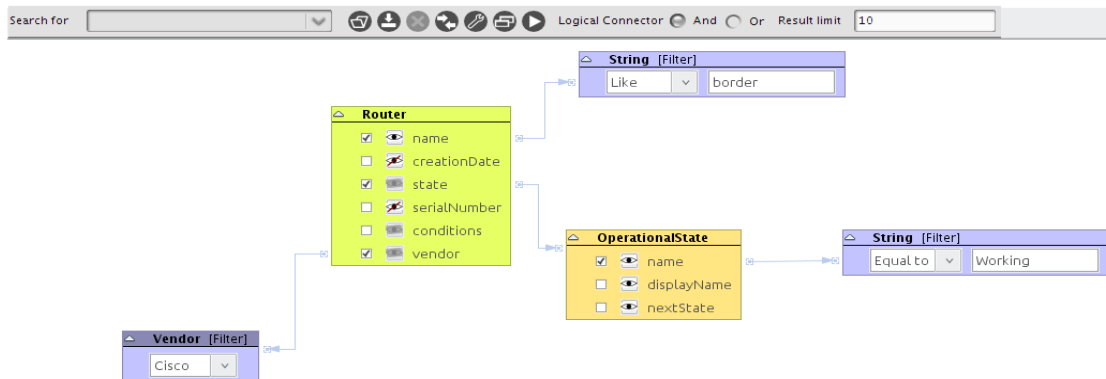


Figure 44: Complex query using an AND logical connector

Please note that the condition “Equal to” is case sensitive, while “Like” it is not.

Query Results

	Router.creationDate	Router.serialNumber	OperationalState.name
BorderRouter_1 [Router]	Sun Mar 13 17:16:15 GMT-05:00 2011	BR987-654	Working

Figure 45: Result set for the complex query

If you hit the “Next Page” button () you will get the second result and so on. The “Show all” button () will retrieve the full list of results.

Exporting results

You can export the results in two formats: CSV and XML. This is done by pressing the Export icon (), which will open a window to choose the location of the output file and the format:

Figure 46: Search results export options

Output file contains the destination (once you choose a directory with the file chooser, a random name will be generated, but you can change it as you wish). Note that writing file the extension is not mandatory, since it will detect the type and append the extension before to save the file.

Export format to let's you choose the output format. The configure button () lets you set some particular options regarding to the selected format. Currently only CSV has advanced options:

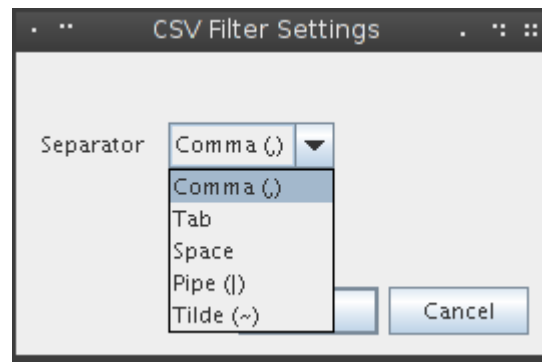


Figure 47: CSV filter settings

There you can choose the separator character between fields. The available options are: Comma(,), Tab (), Space (), Pipe (|) and Tilde (~).

Finally *Range* is used to specify whether whole results should be exported or only the current page.

Note: The XML structure used to export the results can be checked [here](#)[6]

Saving and restoring queries

The queries can be saved as public or private. It's probable that someone wants to execute the same query on a regular basis and if that's the case for many users, he/she would want to let other users to use the such query. After you design a query, you can save it and this small window will prompt for some basic information:

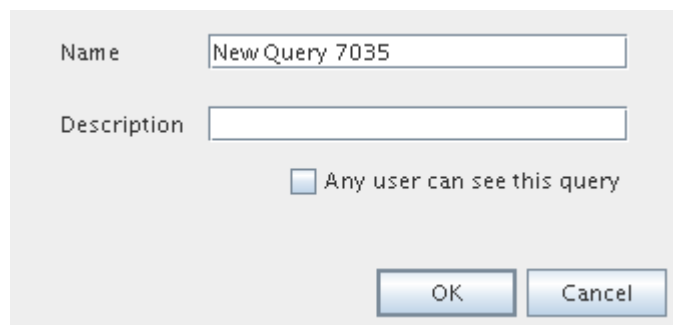


Figure 48: Setting the query metadata before to save it

Name is the query name.

Description is the query description

The last check box lets you to make the current query public. By default, the query is private, so you don't pollute the list of available queries for the rest of the users.

You can change these settings anytime using the configure button().

To open a query, just use the open button (). Then you will be asked if you want to see only your queries or both your queries and the public ones:

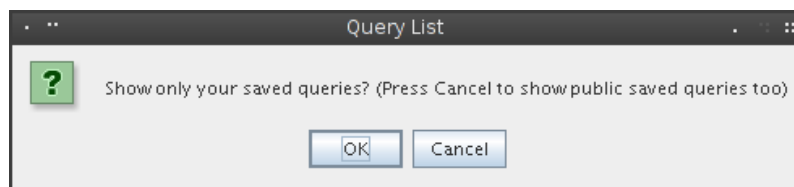


Figure 49: Dialog to choose what queries should be listed

Answering “Cancel” will show you both. “OK” will list only those owned by you.

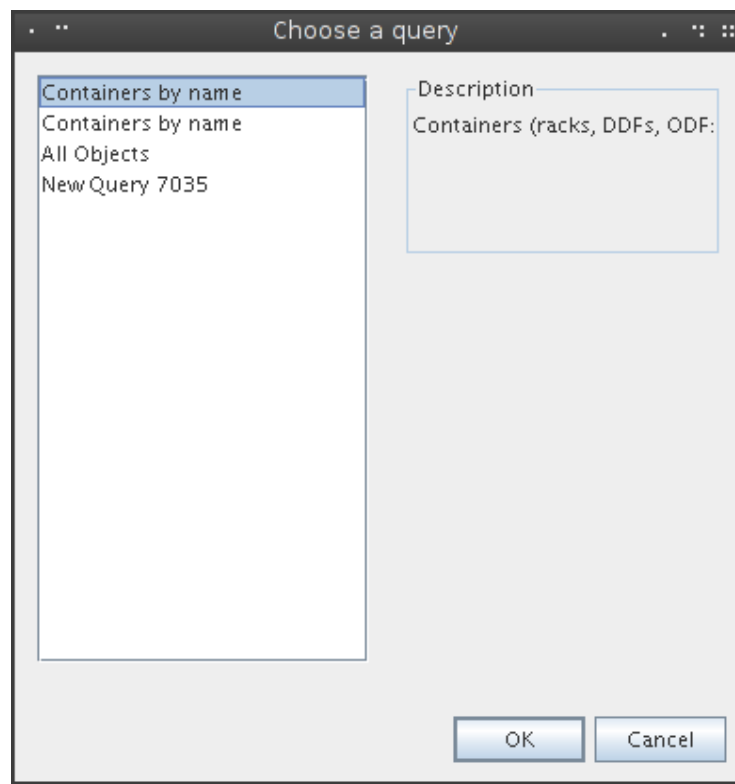


Figure 50: List of available queries

Now you just have to select the one you need and you're done!

Important: Please note that when a query is saved the values used to filter won't be saved, since the filter nodes are used as open parameters. For example:

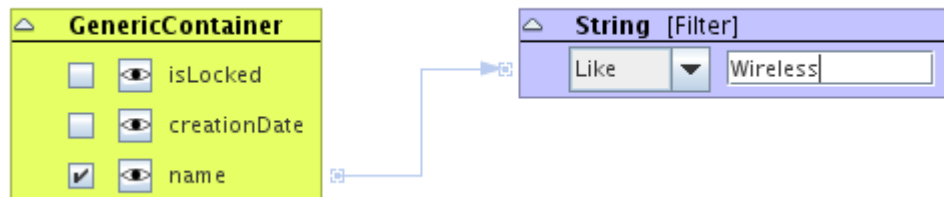


Figure 51: Example of a parametrized query

In the query above, the value “Wireless” won't be saved anywhere. Just the structure and the condition (“like”, in this case). The blue box is a parameter. It's supposed that you will fill it with the value you need depending on what you're looking for.

Note 1: Expanding a list type and not selecting anything from it has the same effect as selecting the value “None” (null) in the compact view of the filter.

Note 2: Every time you run a query, a new window with the results will be opened, so don't forget to close those you don't need to save memory.

Chapter 9. Pools Manager

This module enables you to manipulate pools of objects. Pools are entities where objects that can't be placed in the standard navigation tree are put. Most of them are logical or administrative elements such as VLANs or IP addresses/subnetworks. You can see a pool like a bag where you put things you don't know where else to put. To use this module, click on the icon  in the main toolbar or go to the menu item Tools → Pools. It will open a navigation tree similar to the one used to browse through the containment hierarchy. The first time it will be empty, you can add new pools by right-clicking the root new and selecting “New Pool”, as show in the figure below.

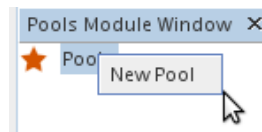


Figure 52: Creating a new pool

The dialog box will prompt you for the newly created pool name, its description and what kind of objects do you want to store inside. If you choose, say, “Router”, it will let you store only instances of Router. On the other hand, if you choose an abstract class (this is any starting with “Generic” or one of the core classes like InventoryObject or ViewableObject) you will be able to place instances of any subclass of theirs.

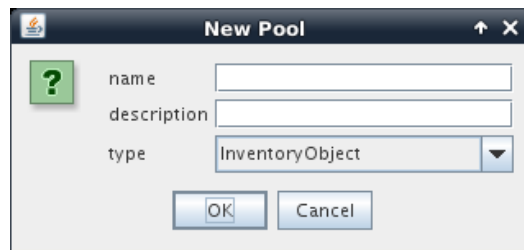


Figure 53: Pool creation dialog

Once it's created, you can't edit its attributes, only add children to it. By right-clicking the corresponding node you can access to all actions associated to it. From that moment on, it behaves exactly like the Navigation Tree and you may have multiple containment levels as shown in the figure below.

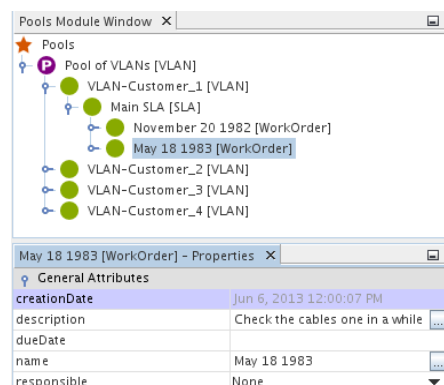


Figure 54: Multiple containment levels

Chapter 10. Rack View

The Rack View allows you to see how elements are distributed in a rack and what's the current usage percentage based on some attributes you must set before to build the view. To activate this view, just open the module and select a rack from the navigation tree, or, in fact, from any tree. This view is only available to instances of class Router. This view depends on the following attributes (all of them case-sensitive):

rackUnits: This attribute **must** exist in all the classes whose instances will be contained by a rack, and have a value different than 0 or null. The default data model **does not** include this attribute in any class other than Rack itself, so you must create it using the Data Model Manager in all classes whose instances will be set as possible children of Rack in the containment hierarchy. *rackUnits* expresses the number of rack units of a Rack or the number or RU that an equipment occupies. The type of this attribute must be Integer. If you try to build this view and inside the rack there's an object that doesn't have this attribute, or it is set to 0 or null, you will get the following error:

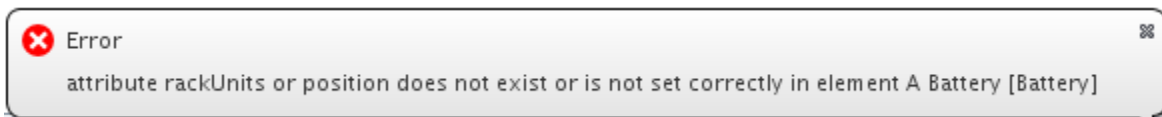


Figure 55: Error produced when one of the children of the rack doesn't have the attribute *rackUnits* or it is null or 0.

The same way, if your rack doesn't have properly set the property *rackUnits*, you'll get a message like this:



Figure 56: Error message displayed if the rack doesn't have properly set its property *rackUnits*

In the former case, just create the attribute in the class whose instance is complaining. Remember that the name must be *rackUnits* and be an Integer:

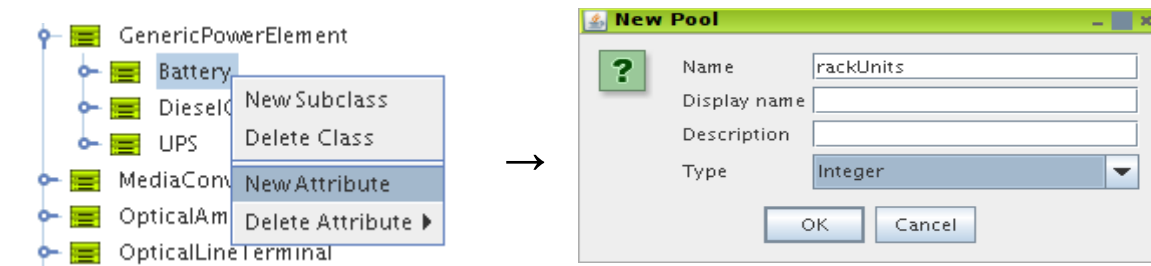


Figure 57: Creating the attribute *rackUnits* in a class whose instances will be contained by a rack

Chapter 11. Service Manager

This is the first appearance of the Service Manager in Kuwaiba. You can expect many changes in the future, extending its capabilities, but in the end, it will always come down to the same: Customers have Services and Services use Resources (physical, logical and administrative). So let's see how this works in the actual application:

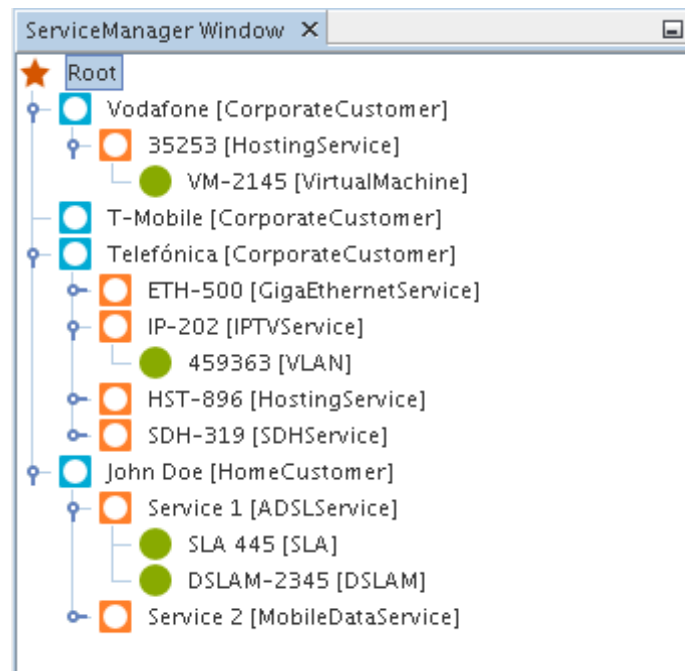


Figure 59: Service Manager

What you see above, is a sample Service Manager tree. You should first create your customers and services by right-clicking on the respective parent nodes (the root node for Customers and each Customer to every Service). Once you've created that structure, you are ready to start associating resources.

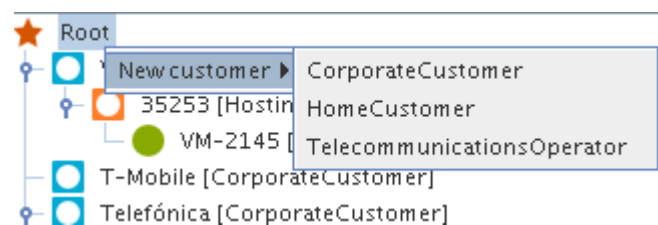


Figure 60: Adding a customer

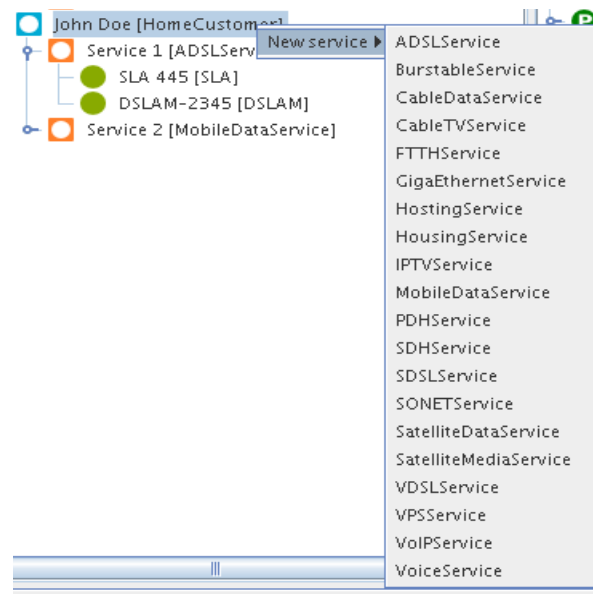


Figure 61: Adding a service

To add resources to an existing service, you just have to locate it in the Navigation Tree (or any other tree), or search for it using the query manager. Then right click on it and select “Relate to service” as shown below:

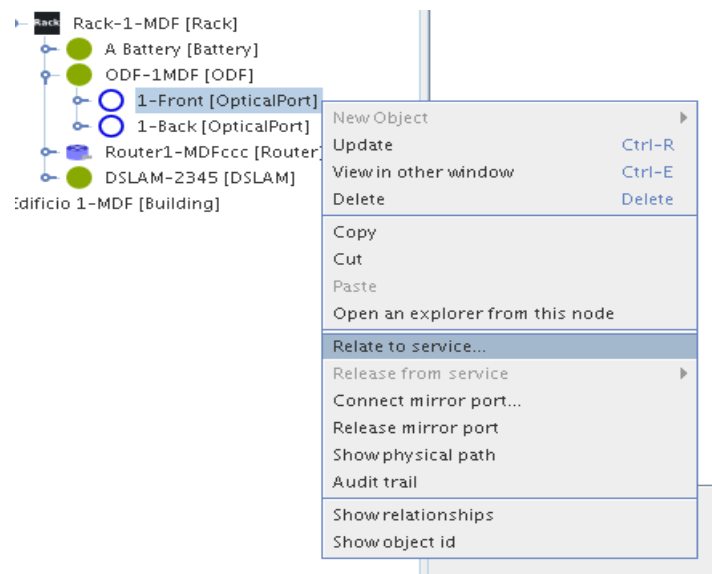


Figure 62: Relating an object to a service from the Navigation Tree

You can associate a single resource to many services.

You can also do this with wire containers or fibers or cables in the Object View module.

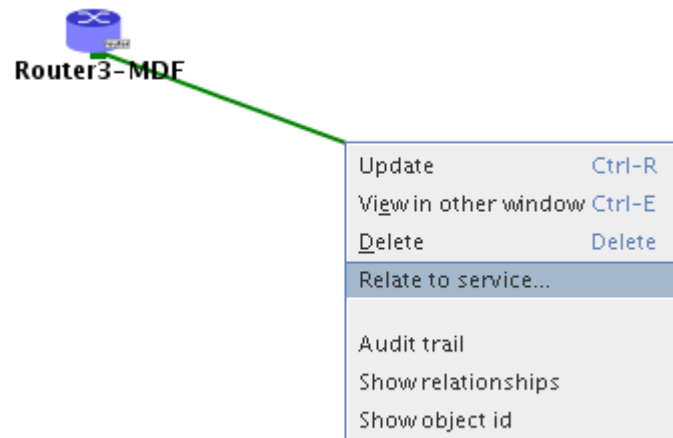


Figure 63: Relating a fiber to a service from the Object View

Once you've selected this option, a list of available services will be shown so you can choose the service to be related to the object. The system will warn you if the objects are already related:

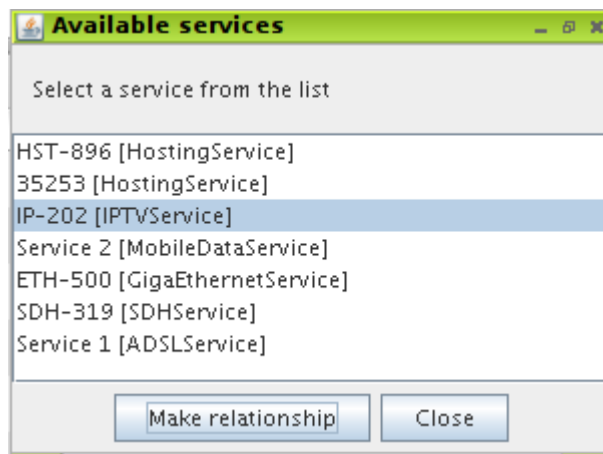


Figure 64: List of available services

You can see now, under the service, the list of related resources. If the Service Manager is

open, you can refresh it using the  button in the main toolbar. If you want to see what services are using a given object, just select the “Show relationships” option from the contextual menu of every object and expand the “uses” node:

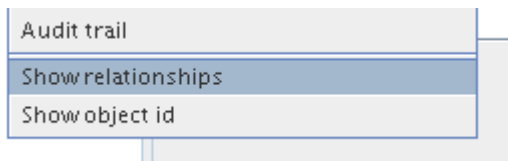


Figure 65: Showing relevant relationships

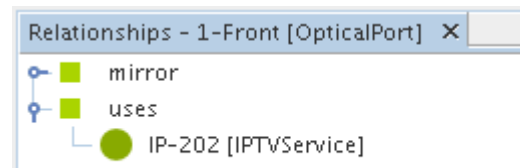


Figure 66: List of services using this object

You can easily unrelate an object from a service by selecting the “Release from service” option in the contextual menu and clicking on the desired service:



Figure 67: Unrelating an object from a service

Chapter 12. User and Group Management

This feature lets you define new application users and relate/unrelate them from application groups. By default there is only one user (**admin**, password **kuwaiba**) and two groups: admins and users. You can edit everything, even its username.

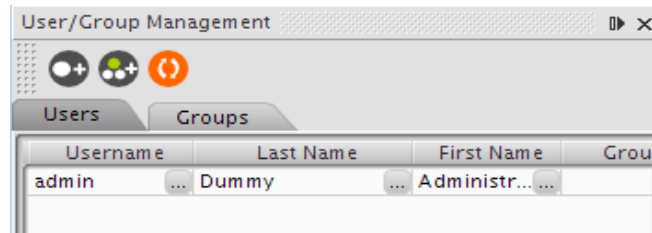


Figure 68: User/Group Management main window

The module shows three buttons at its own toolbar:

- **Add user:** Adds a new user. It's placed at the end of the list and its name is a composition of the string “user” and a random number between 0 and 10000.
- **Add groups:** Adds a new group. It's placed at the end of the list and its name is a composition of the string “group” and a random number between 0 and 10000.
- **Refresh:** Refreshes the selected tab (users or groups)

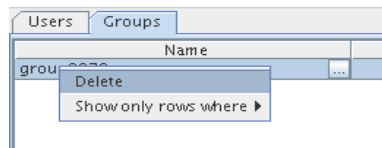


Figure 69: Deleting a group

If you want to delete an user or a group, just right click the element and choose “Delete”. Deleting a group won't delete the related users.

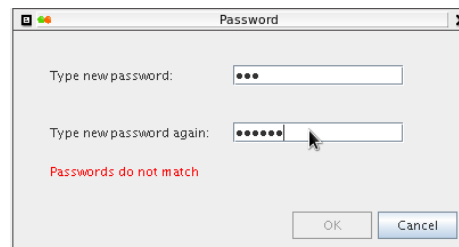


Figure 70: Changing a password

If you want to change a user's password, click the  button in the “Password” or “Group” column and a custom editor will be shown.

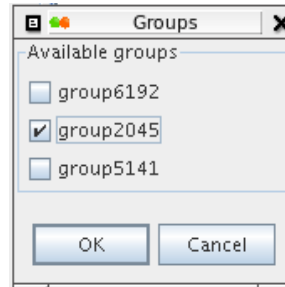


Figure 71: Changing an user group association

Currently is only possible to add/remove groups to/from a user (not vice versa)

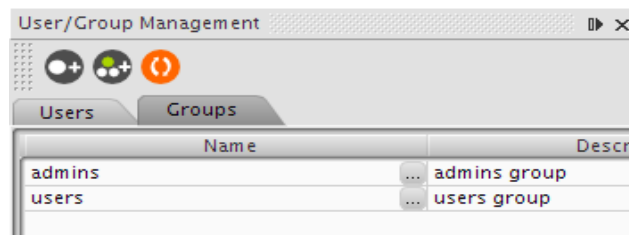


Figure 72: Group list

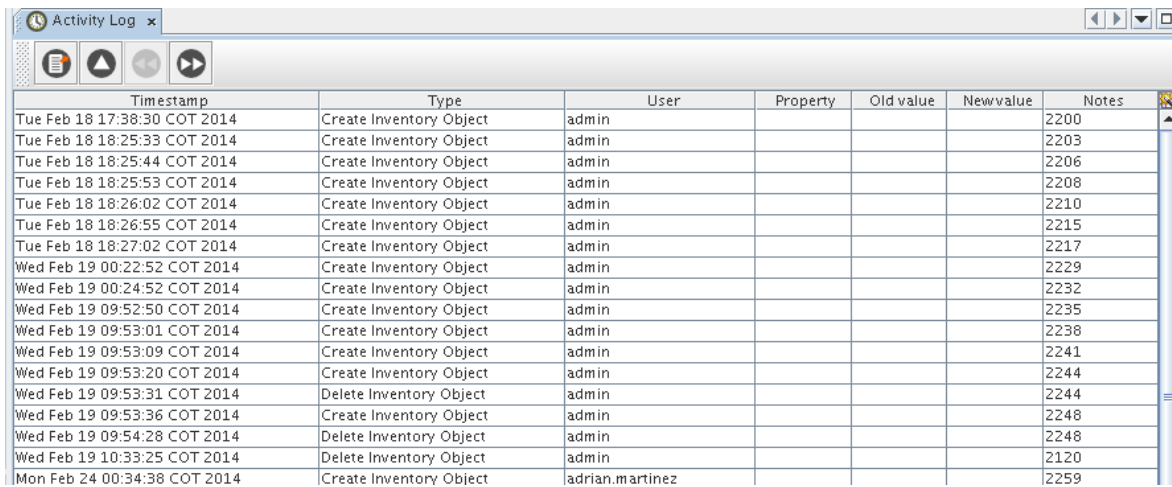
Shortcut: CTRL+U

Hint: Selecting any node in the tree and writing a text string will attempt to search for a user name with such string. If there are several matches, F3 will jump to the next one.

Chapter 13. Audit Trail

Kuwaiba saves every update in the inventory. The logs might refer to an element created or removed or might be related to the update of an existing object. To see the list of

Create/Delete actions, click on the button  in the main toolbar. A table with 7 columns will be displayed:



Timestamp	Type	User	Property	Old value	New value	Notes
Tue Feb 18 17:38:30 COT 2014	Create Inventory Object	admin				2200
Tue Feb 18 18:25:33 COT 2014	Create Inventory Object	admin				2203
Tue Feb 18 18:25:44 COT 2014	Create Inventory Object	admin				2206
Tue Feb 18 18:25:53 COT 2014	Create Inventory Object	admin				2208
Tue Feb 18 18:26:02 COT 2014	Create Inventory Object	admin				2210
Tue Feb 18 18:26:55 COT 2014	Create Inventory Object	admin				2215
Tue Feb 18 18:27:02 COT 2014	Create Inventory Object	admin				2217
Wed Feb 19 00:22:52 COT 2014	Create Inventory Object	admin				2229
Wed Feb 19 00:24:52 COT 2014	Create Inventory Object	admin				2232
Wed Feb 19 09:52:50 COT 2014	Create Inventory Object	admin				2235
Wed Feb 19 09:53:01 COT 2014	Create Inventory Object	admin				2238
Wed Feb 19 09:53:09 COT 2014	Create Inventory Object	admin				2241
Wed Feb 19 09:53:20 COT 2014	Create Inventory Object	admin				2244
Wed Feb 19 09:53:31 COT 2014	Delete Inventory Object	admin				2244
Wed Feb 19 09:53:36 COT 2014	Create Inventory Object	admin				2248
Wed Feb 19 09:54:28 COT 2014	Delete Inventory Object	admin				2248
Wed Feb 19 10:33:25 COT 2014	Delete Inventory Object	admin				2120
Mon Feb 24 00:34:38 COT 2014	Create Inventory Object	adrian.martinez				2259

Figure 73: General actions Audit Trail

The first column is the timestamp of the action. The second is the type of action, which, currently can only be “Create Inventory Object” or “Delete Inventory Object”. The third is the user who performed the action. The fourth is the affected property, that will always be empty in this table because all records refer either to a creation or to a deletion events. For the same reason the fifth and sixth columns will be empty, and the last column, “notes”, is set to the id of the object that was created or deleted.

Using the buttons in the top toolbar you can jump from page to page (each one has 100 records) or export the data into CSV or XML files. If you want to see the activity logs related to a particular object (that is, the changes performed on it from its creation), just right click on the element and select Audit Trail. You will see the same structure, but the three central columns will be filled:

Audit trail for ODF2-EQ [ODF]						
Timestamp	Type	User	Property	Old value	Newvalue	
Fri Feb 07 14:00:47 COT 2014	Move object	admin	parent	1770	1765	
Fri Feb 07 14:01:12 COT 2014	Update Inventory Object	admin	name		ODF2-EQ	
Mon Feb 24 05:47:57 COT 2014	Update Inventory Object	adrian.martinez	rackUnits		5	
Mon Feb 24 05:48:00 COT 2014	Update Inventory Object	adrian.martinez	rackUnits	5	42	
Mon Feb 24 05:48:08 COT 2014	Update Inventory Object	adrian.martinez	vendor		Techtronix	
Mon Feb 24 05:48:15 COT 2014	Update Inventory Object	adrian.martinez	position		5	
Mon Feb 24 05:49:02 COT 2014	Update Inventory Object	adrian.martinez	state		Working	
Mon Feb 24 05:49:04 COT 2014	Update Inventory Object	adrian.martinez	state	Working	Not Working	

Figure 74: Audit trail for a particular object

You can also export these results to CSV or XML files. The records are read only. Most of the recorded actions in this case are attribute updates and changes of parent (when an object is moved from one place to another).

Resources

- [1] Definition of Asset. <http://en.wikipedia.org/wiki/Asset>
- [2] Overview of ITIL. <http://en.wikipedia.org/wiki/ITIL>
- [3] Thread to request changes in the data model
<https://sourceforge.net/projects/kuwaiba/forums/forum/1129209/topic/3734242>
- [4] Client installation instructions http://kuwaiba.sourceforge.net/index.php?option=com_content&view=article&id=58:client-instructions&catid=38:user-documentation&Itemid=55
- [5] Object Oriented Programming. http://en.wikipedia.org/wiki/Object-oriented_programming
- [6] XML in Kuwaiba. https://sourceforge.net/apps/mediawiki/kuwaiba/index.php?title=XML_Documents
- [7] Class hierarchy <http://kuwaiba.sourceforge.net/javadoc/server/ejb/overview-tree.html>
- [8] Class inheritance http://en.wikipedia.org/wiki/Inheritance_%28object-oriented_programming%29
- [9] Picture showing the data model as a class hierarchy tree
<http://kuwaiba.neotropic.co/images/dataModel.jpg>